US005742237A

United States Patent [19]

Bledsoe

[11] Patent Number: 5,742,237

[45] Date of Patent: Apr. 21, 1998

[54] **TAG LOCATION MONITOR**

[75] Inventor: **William Byron Bledsoe**, Marietta, Ga.

[73] Assignee: **Lockheed Martin Corporation**, Bethesda, Md.

[21] Appl. No.: **565,092**

[22] Filed: **Nov. 30, 1995**

[51] **Int. Cl.**[6] **G06K 15/00**; H04B 7/00; G08B 5/22; H04Q 7/00

[52] **U.S. Cl.** **340/825.49**; 340/825.35; 340/825.54; 340/572; 235/375

[58] **Field of Search** 340/825.49, 825.54, 340/825.35, 825.02, 825.06, 539, 572, 825.03, 827, 825.36, 505, 568; 235/375

[56] **References Cited**

U.S. PATENT DOCUMENTS

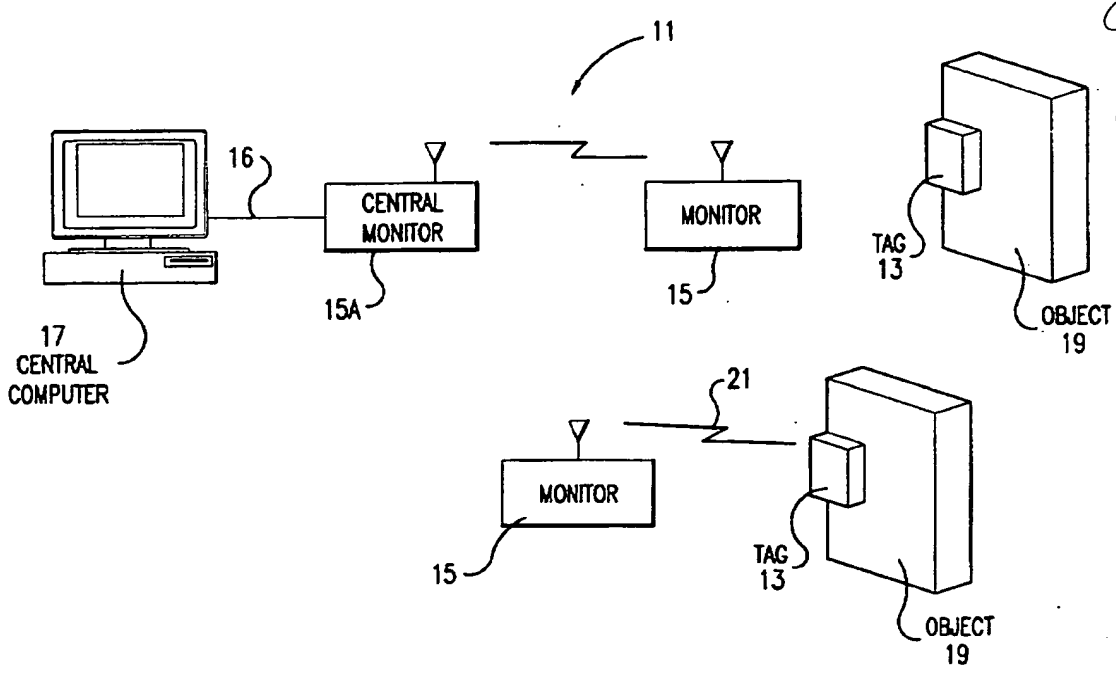
4,783,655	11/1988	Cobb et al.	340/825.49
4,814,742	3/1989	Morita et al.	340/825.54
4,816,824	3/1989	Katz et al.	340/825.34
5,051,741	9/1991	Wesby	340/825.49
5,401,947	3/1995	Poland	235/383
5,424,858	6/1995	Gillotte	359/143
5,442,343	8/1995	Cato et al.	340/825.35
5,448,226	9/1995	Failing, Jr. et al.	340/825.35
5,537,126	7/1996	Kayser et al.	345/1
5,565,858	10/1996	Guthrie	340/825.35
5,594,425	1/1997	Ladner et al.	340/825.06

Primary Examiner—Michael Horabik
Assistant Examiner—Yonel Beaulieu
Attorney, Agent, or Firm—Eric R. Katz

[57] **ABSTRACT**

A monitor network of a tag location system includes a grid of monitors for transmitting and receiving messages from neighboring monitors and tags within range of the monitor. Also provided is a memory device for storing: a unique identity of the monitor; each input partner monitor from which the monitor receives messages, and an output partner monitor to which the monitor sends messages; monitor and tag signals; a signal strength of the tag signals and significant changes thereto; an acknowledgement signal; and a distress signal indicating that the monitor has not received acknowledgement signal from its output partner monitor. Further there is a computer for periodically causing all tag signals, associated signal strengths and significant changes stored in the memory device to be transmitted to the output partner monitor, for forwarding received monitor messages to the output partner monitor, for issuing the acknowledgement message to be transmitted of a monitor message, for determining receipt of an acknowledgement and for sending a distress signal to all neighboring monitors within range of the monitor if an acknowledgement signal is not received from the output partner monitor. Tag signals issued by tags and received by the monitor directly from tags within range of the monitor or from input partner monitors are transmitted to the output partner monitor for forwarding to the central station, and wherein, monitor messages issued by the central station and receive from input monitors are transmitted, if required, to the output monitor.

12 Claims, 90 Drawing Sheets



US005804810A

United States Patent [19]

Woolley et al.

[11] Patent Number: 5,804,810

[45] Date of Patent: Sep. 8, 1998

[54] **COMMUNICATING WITH ELECTRONIC TAGS**

[75] Inventors: **Louis A. Woolley**, Clinton; **James H. Weimar**, Minoa, both of N.Y.

[73] Assignee: **Par Government Systems Corporation**, New Hartford, N.Y.

[21] Appl. No.: **672,342**

[22] Filed: **Jun. 26, 1996**

[51] **Int. Cl.**[6] **G06K 19/06**; G06K 15/00; G06F 17/00

[52] **U.S. Cl.** **235/492**; 235/375; 235/383

[58] **Field of Search** 235/383, 492, 235/375, 382, 486, 385, 380; 340/825.35, 825.31, 572; 342/44

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,688,244	8/1987	Hannon et al.	379/58
4,750,197	6/1988	Denekamp et al.	379/58
5,173,594	12/1992	McClure	235/375
5,223,844	6/1993	Mansell et al.	342/357
5,347,274	9/1994	Hassett	340/988
5,565,858	10/1996	Guthrie	340/285
5,627,517	5/1997	Theimer et al.	340/572

FOREIGN PATENT DOCUMENTS

0268390	11/1990	Japan	235/486

OTHER PUBLICATIONS

"Radio tracking/ID & security system", Savi Technology, Commercial Carrier Journal, p. 122, Aug. 1995.

The Savi Asset Management System[198], Savi Technology, Inc., Palo Alto, CA.

TransitMaster™ An Integrated, GPS–Based Information Management System for Transit, Rockwell International, Cedar Rapids, IA, Aug. 1995.

"TMS Wins Vehicle Location Contract for Des Moines Metro", Earth Observation Magazine, p. 12, Oct. 1995.

"Will TI make a Bundle on 'Smart' packages?", Business Week, p. 112, Nov. 1995.

ProfitMAX™ Highlights, Integrated Cargo Management Systems, Inc., 3 pages.

"Auto–Trac Provides the Information You Need", Auto–Trac, Dallas, TX, 4 pages.

3M Fleet Management System Pinpoints Bus Activity, 3M, 2 pages.

Primary Examiner—Donald T. Hajec
Assistant Examiner—Thien Minh Le
Attorney, Agent, or Firm—Fish & Richardson P.C.

[57] **ABSTRACT**

An object in a storage area or moving vehicle is monitored by attaching an electronic tag to the object. An electronic device detects the presence of the object by communicating with the tag while the object is in storage or is being moved by the vehicle. The tags may also determine the location of an attached object and may reroute the object if it deviates from a given shipping schedule. A group of objects is monitored by two electronic tags, each attached to an object in the group. Each tag has circuitry for communicating information relating to an object in the group to a second tag. Each tag also includes a memory connected to the circuitry that is capable of storing the information, and a controller connected to the memory and the circuitry. A computer is used that has circuitry for communicating with a tag.

23 Claims, 59 Drawing Sheets

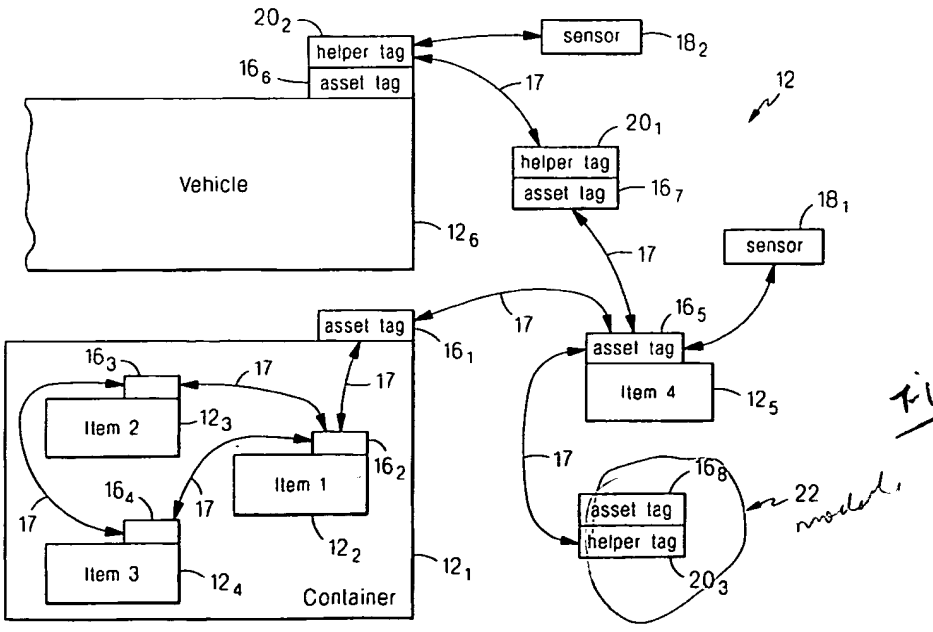


Fig. 12, 15

The tag also notifies the joining unit of the detachment in step **464**. Alternatively, the subfunction may cause the router to notify a device, such as a user-operated device, instead of a gateway of the detachment.

(b) Joining

The joining functional unit **372** includes an existence subfunction **470** connected below a hierarchy subfunction **472** (FIG. **24**).

Referring to FIG. **29**, when the attachment subfunction indicates to the existence subfunction that the tag is active (step **480**), the existence subfunction causes the tag to look for other tags by attempting to initiate communications through the router on a particular hailing channel (**482**). If other tags exist (step **484**), the subfunction notifies the hierarchy and supplies the identity codes of the tags (step **486**).

If a group of tags are located near the asset tag, the subfunction chooses one asset tag (a netmaster) to perform some of the location and network formation units. This prevents the tags from duplicating calculations already performed by other tags.

Referring to FIG. **30**, the hierarchy subfunction communicates with the other tags to determine if the other tags have designated a particular tag as netmaster (step **487**). If the tag locates a netmaster (step **488**), it notifies the network formation unit through the location unit (step **489**). Otherwise, the tag chooses the tag with the highest identification number as the netmaster, and chooses the tag with the next highest identification number as the assistant netmaster (step **490**).

The netmaster periodically surveys the tags for new or missing members, and notifies the hierarchy subfunction. The netmaster uses a particular hailing frequency for its communications.

If the netmaster disappears or indicates its battery is low, the assistant netmaster replaces the netmaster, selects another assistant, and informs the hierarchy subfunction of the change.

(c) Location

Referring to FIG. **25**, the location unit **374** determines the locations of tags in the network using the following subfunctions: intertag distance **500**, verify range **502**, refine range **504**, location compute **506** and detect change **508** (FIG. **25**).

Referring to FIG. **31**, the intertag distance subfunction **500** determines the distance and the covariance (or measurement error) of the distance between the tag and each tag identified by the joining unit (step **530**). The subfunction also communicates with the tags to obtain location information for the tags (step **532**). The location information is typically obtained from a GPS sensor located near the tags. The subfunction organizes this information in the tag memory (step **534**), and notifies the verify range subfunction **502** that the information is available (step **536**).

The verify range subfunction verifies these ranges, and eliminates ranges that are clearly erroneous. The refine range subfunction provides accurate estimates for the ranges eliminated in the previous subfunction. The compute locations subfunction transforms the corrected ranges supplied by the refine range function into an arbitrary set of coordinates for each tag. Using the location information, the location subfunction transforms the arbitrary coordinate system into an absolute coordinate system, and determines a location for the tag.

The detect change subfunction receives the location information from the verify range subfunction, and monitors the ranges of other tags in the network to detect a change in the tag's location.

Each of these subfunctions is discussed in more detail below.

(d) Network Formation

The network formation unit includes a determine topology subfunction **560**, a sentinel subfunction **562** and a message subfunction **564** (FIG. **25**). The determine topology subfunction receives information from the change detection subfunction indicating changes in the existence of a tag, and the location of the tag over time. Using this information, the subfunction determines a set of neighbors for each tag. A topology description of the set is found using a Delaunay triangulation, described below, which describes the topology as a convex hull and identifies the set of nearest neighbors. When the determine topology subfunction receives updated information from the location unit, it modifies the topology description.

The sentinel subfunction identifies tags that are located on the convex hull. These tags (referred to as sentinels) are assigned to communicate with tags outside the hull. The subfunction also identifies gateway tags capable of long distance communication and PDAs acting as a man-machine interface. The subfunction stores the identity of each gateway in the gateway table.

The sentinel subfunction communicates the identity of the sentinels to the hierarchy subfunction **472** (FIG. **24**) over the feedback path **380**. The hierarchy subfunction causes the sentinel to search for new tags, extending the range of the netmaster.

The sentinel subfunction also responds to commands received over the router **382** (FIG. **23**). Each time a command from an entity is routed to the sentinel subfunction, the subfunction first verifies that the source of the message is listed in its entity table. If the entity is not listed, the subfunction ignores the message. If it is listed, the unit responds to the command.

The first command received by the sentinel subfunction queries whether a tag with a given identification exists. The subfunction checks whether any stored identification matches any stored tag's identification number. If it does, it answers "yes" to the entity. The unit then checks whether the entity is listed as a gateway in the gateway table. If it is not listed, the unit records the identity of the entity as a gateway. If there is insufficient memory to store the identity in the table, the unit communicates with the oldest recorded gateway in the list. If it is unable to establish communication, it deletes the gateway from the list. The unit repeats this for each gateway in the list.

The second command asks whether there are tags which have not been queried by a gateway. The subfunction looks at its gateway-table, and if the gateway ID is not listed, the unit replies with the tag ID.

The last command asks whether a tag with a particular ID has registered the gateway. If the tag ID is the same as that in the message, the unit checks its entity and gateway table and records the ID of the gateway if it is not already present in the table. If there is insufficient memory to store the gateway, the tag repeats the process described above to delete non-communicating gateways.

The propagate message subfunction defines a network containing a set of efficient communication links between tags using the topology derived by the topology subfunction. The subfunction accomplishes this by assigning tags in the networks as relay points for communications between a pair of tags.

The subfunction also chooses a second netmaster based on the topology. The second netmaster redefines the network whenever a tag is added or removed from the network, and

communicates information regarding the changes in the network to the gateways listed in the gateway table. In some embodiments, the netmaster only notifies the gateway of changes regarding specific tags. The other tags in the network do not duplicate the functions of the netmaster.

The netmaster polls members of the network periodically to make sure they still exist. Each tag in the network checks the time interval since it was last polled by the netmaster. If the time interval exceeds a threshold, the network formation unit in the tag removes the tag from the network.

When the network grows too large, the network formation subfunction divides the network into smaller subnetworks. The subfunction designates channels of communication within a subnetwork that do not interfere with communications in the other subnetwork. In addition, communications within a subnetwork are coded in compacted form. For example, the subfunction assigns each tag a truncated ID. If spread spectrum communications are used, the subfunction uses a short code for communication relayed between tags.

The subfunction designates a tag for maintaining communication with another subnetwork. When the subnetwork grows too small, the subfunction queries the tag for the existence of the other subnetwork. If the subnetwork still exists, the subfunction merges the two subnetworks.

(e) Network Use

The network use unit **378** includes a manager subfunction **590**, an agent subfunction **592**, an inventory subfunction **594**, and a read and write subfunction **596** (FIG. **25**).

Referring to FIG. **32**, the manager subfunction continually checks whether route information is available in the network (step **608**). The information is typically provided by a monitoring device, computer or operations center.

When the information is available, the subfunction stores information regarding the transportation route scheduled for an asset attached to the tag and a checking frequency (step **610**). The subfunction then surveys the network to determine whether there has been an update to the route information (step **611**). If there is, the subfunction returns to step **610** and stores the new information.

If no update information is available, the subfunction examines the location of the tag (as supplied by the location unit) (step **612**), and compares the location to the route (step **614**). If the location of the asset deviates from the route (step **616**), the subfunction informs a gateway through the network (step **618**) and returns to step **611**. If no deviation exists, the subfunction repeats the process by returning to step **611**.

As shown in FIG. **33**, the agent subfunction **592** stores information regarding the destination of the attached asset (step **630**). When a deviation from the route is reported by the manager subfunction (step **632**), the agent subfunction queries the network regarding available alternative routes (step **634**). If a more effective alternative route is available (step **636**), the subfunction notifies a gateway (step **638**).

In some embodiments, the route includes a series of locations where the conveyance carrying the asset is changed (transhipment points). The subfunction seeks a conveyance for the next part of the route at each transhipment point. The subfunction also searches for other tags following a similar route, and searches for a proper conveyance for carrying all the assets together. The subfunction then informs a tag on the desired conveyance that the conveyance should load the assets. The subfunction thus performs functions analogous to a node in a packet switched network.

The inventory subfunction maintains a list of assets in the network. This subfunction is particularly useful in a warehouse application.

The read and write subfunction stores and retrieves data from the asset tag's memory. The subfunction is activated by the following commands from one or more entities stored in the entity list.

1. Read data to one or more identified entities.
2. Clear data as instructed by one or more identified entities, and write certain data.
3. Append the following data as instructed by one or more identified entities.
4. Add an identified entity to the entity table as instructed by one or more entities.
5. Delete an entity from the entity table.
6. Read the entity table.
7. Read the gateway table.

X. Detailed design of the location unit

A. Intertag Distances Subfunction

(a) Baseline Method.

The intertag distances subfunction (FIG. **34**) in a tag **640** determines the tag's distance, that is, range, from another tag **642** by measuring the time required for an electromagnetic pulse to travel round-trip between the two tags. There are several methods for measuring this time. A baseline method is described first, followed by descriptions of alternate methods. Most of the methods require that the pulse-originator tag have a cooperating transponder tag for a reference, with one clock on each tag. However, the last alternate method describes a dual-clock technique wherein both clocks are on the originator tag—the other tag is simply a repeater or a repeater with a delay. In fact, if a delay is not required, another active tag is completely unnecessary in the last alternate method; the reference can be totally passive, for example, a mirror or a three-dimensional corner reflector.

The resolution of the range measurement is proportional to the time-measuring resolution of the tag, so it is advantageous to use methods of range measurement that effectively increase the time-measuring resolution. The methods described below take advantage of the tag's ability to detect a pulse received during a period of the tag's time-measuring clock. For a single pulse, the tag can detect that the pulse arrived during the period but not when during the period. Therefore, the range-measurement methods use multiple pulses and multiple successive measurements. In the baseline method, small offset delays of differing but known lengths are inserted, one per round-trip, in the round-trips of the pulses, to stagger the arrivals of the pulses. Thus, for a set of ranges, the offset delays cause a fraction of the pulses to return in a later time-measuring clock period in any particular range measurement. As described below, the method determines the range from this fraction.

In the baseline method, the originator tag and the transponder tag initially have identical clock periods. To insert a multiple of an offset delay "d" in the round-trips (FIG. **35**), the originator alters its clock period slightly. The length of the alteration is an amount of time which provides a one-clock-period difference with the transponder's clock over a measurement interval. The measurement interval is an amount of time equal to the number of pulses sent, "N," multiplied by the time interval between the sending of each pulse. Although the originator's clock may be altered above or below the frequency of the transponder's clock, the embodiment of the baseline method described herein uses an originator's clock with a higher frequency, that is, a shorter period, than the transponder's clock.

The difference in clock periods results in the insertion of differing but known delays in the round-trip time, in the following way. Like the originator tag, the transponder tag

The tag then decides whether the tag itself is moving, a reference tag is moving, or both. If the tag itself is moving, the tag determines its instantaneous position, and velocity using the relative position information. If the tag is stationary, the tag converts its relative position to an absolute position.

The first condition is that the reference tags must be moving in three dimensional space relative to the tag.

If the tag and reference tags are coplanar but not co-linear, the reference tags must be moving in two dimensions relative to the tag. If the tag and reference tags are co-linear, the reference tags must be moving in one dimension relative to the tag.

As additional reference positions and errors are accumulated by the intertag distance subfunction, the tag refines its position estimate using the same method. In addition, as tag locations are identified and changed, this information is passed to the determine topology subfunction.

XI. Detailed description of network formation unit

A. Determine topology subfunction

The determine topology function designates a set of neighbors for each tag by means of Delaunay triangulation, described in detail in "Spatial Tessellations: Concepts and Applications of Voronoi Diagrams," A. Okabe et al, the contents of which are incorporated herein, in its entirety, by reference.

Referring to FIG. **53**, the neighbors of a tag 16_1 are found by first locating the set of points that are closer (in Euclidean distance) to tag 16_1 than to any other tag. Each set of points is bounded by an "ordinary Voronoi polygon" **2000**. The neighbors of each tag include the tags in polygons adjacent to polygon **2000**. Thus, the neighbors of tag 16_1 consist of tags $16_2, \ldots, 16_8$.

Referring to FIG. **54**, a preliminary communication link **2010** is defined between a tag and each neighboring tag. The set of links between tags forms a Delaunay triangulation, and defines the topology of the set of tags.

The Delaunay triangulation is defined incrementally by causing each tag to find its neighbors. If the tags are all activated at the same time, the netmaster (the tag with the highest ID number) finds its neighbors first, by using the algorithm described below. After the netmaster has completed the algorithm, it selects the tag having the second highest ID number, and commands the tag to perform the algorithm to find its neighbors. The netmaster continues selecting tags in decreasing order of ID number until all tags have been selected.

When a new tag is added to a network, it communicates with other tags to determine whether the tags have already performed the process described above, that is, whether a Delaunay triangulation for the tags has been determined. If it has, links must be added to the triangulation to connect the new tag to its neighbors. This is accomplished by having the new tag perform the algorithm described below. If no Delaunay triangulation exists, the new tag waits until ordered by the netmaster to perform the algorithm.

The new tag (or the tag designated by the netmaster) finds and maintains a list of its neighbors by performing the following algorithm. Each step of the algorithm is executed by the determine topology subfunction in the new tag.

The new tag begins by receiving its own coordinates and then seeks information locating other tags in its vicinity from the compute locations subfunction. The determine topology subfunction ignores the z coordinates in the location information and thus finds the topology of the tags using a two-dimensional projection of their locations.

The determine topology subfunction next selects a random tag from the coordinate information which it defines as the "current tag" (current(n)) where n is the index of the iteration and queries the current tag regarding its neighbors. In other embodiments, the tag may communicate with a sentinel (defined in Section IX.C(c), above) that causes the tag to choose the sentinel as the current tag. Any other tag may be chosen as the current tag.

Referring to FIG. **55**, the new tag calculates its distance from the current tag and its distance from each neighbor of the current tag, using the coordinate information (step **2014**). The new tag analyzes the distances to determine whether the new tag is closest to the current tag (steps **2016, 2018**). If so, the new tag stores the identification number of the current tag in its list of neighbors (step **2020**) and proceeds to step **2022** in FIG. **56**. If not, the new tag chooses the neighbor of the current tag that is nearest the new tag as the current tag (step **2022**) and returns to step **2014**.

When the new tag arrives at step **2022**, the current tag has been defined as either the last current tag or the neighbor of the last current tag that is closest to the new tag. In step **2024** (FIG. **56**), the new tag defines two of the neighbors of the current tag as the "next tag" and the "previous tag."

The next tag of a particular tag A is defined relative to a second tag B. The next tag is the neighbor of tag A which forms the smallest positive angle c with respect to tag B, with the tag A being placed at the vertex of the angle. For example, as shown in FIG. **57**, tag A is the current tag 16_1 and the tag B is the new tag 16_2. If the neighbors of the current tag include tags 16_3, 16_4 and 16_5, then the next tag of the current tag relative to the new tag of the current tag is tag 16_3. Similarly, the next tag relative to tag 16_3 is tag 16_4.

The previous tag of a current tag relative to a second tag is defined as follows. The previous tag is the neighbor of the current tag which forms the smallest negative angle d with respect to the second tag, with current tag placed at the vertex of the angle. In the example shown, the previous tag for the current tag 16_1 with respect to the new tag 16_2 is the tag 16_4. Similarly, tag 16_3 is the previous tag for the current tag relative to tag 16_4.

The new tag defines the next tag (next(n)) and the previous tag (previous(n)) of the current tag relative to the new tag (step **2024**). The new tag then checks whether the angle between the previous tag and the next tag (with the current tag at the vertex of the angle) is less than 180 degrees (step **2026**). For example, in FIG. **57**, the new tag checks whether the angle d is less than 180 degrees. In the same step, the new tag verifies that the previous and next tags are neighbors of one another (that is, they are connected in the Delaunay triangulation). The new tag accomplishes this by querying the previous tag and the next tag regarding their neighbors.

If both conditions of step **2026** are satisfied, the new tag checks whether the new tag is located inside a circle circumscribing the current, next and previous tags (step **2028**). For example, as shown in FIG. **58**, circle **2029** circumscribes tags 16_1, 16_3, 16_4 and contains the new tag 16_2. If the condition of step **2028** is satisfied, a new previous tag (previous (n+1)) is defined as the next tag (next (n)), and a new next tag (next (n+1)) is defined as the tag having the smallest positive angle with respect to the next tag (next (n)). The new tag then increments the value of the iteration (n=n+1) and returns to step **2026**.

If the condition in step **2026** is not satisfied, the new tag checks whether the new tag is between the previous tag and the next tag with respect to the current tag (step **2034**). This means that the previous and next tags are previous and next tags relative to the new tag. For example, in FIG. **57**, the new tag 16_2 is located between the previous tag 16_5 and the next tag 16_3.

US005774876A

United States Patent [19]

Woolley et al.

[11] Patent Number: 5,774,876

[45] Date of Patent: Jun. 30, 1998

[54] MANAGING ASSETS WITH ACTIVE ELECTRONIC TAGS

[75] Inventors: **Louis A. Woolley**, Clinton; **Charles F. Ferrara**, Sauquoit; **Ian Greasley**, Camden; **James H. Weimar**, Minoa, all of N.Y.

[73] Assignee: **Par Government Systems Corporation**, New Hartford, N.Y.

[21] Appl. No.: **671,491**

[22] Filed: **Jun. 26, 1996**

[51] **Int. Cl.**[6] **G06F 17/60**

[52] **U.S. Cl.** **705/28**; 235/385; 340/568; 340/572; 340/825.54; 364/478.01; 364/478.02; 364/478.03; 364/478.13

[58] **Field of Search** 235/375, 385; 340/539, 568, 572.8, 825.54; 342/42; 364/400, 478.01, 478.02, 478.03, 478.13, 478.14, 550; 395/228; 705/28

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,426,326	2/1969	Goldstein	340/825.54 X
4,688,244	8/1987	Flannon et al.	379/58
4,750,197	6/1988	Denekamp et al.	379/58
4,918,425	4/1990	Greenberg et al.	340/539
5,030,807	7/1991	Landt et al.	235/375
5,113,344	5/1992	Kellogg et al.	364/424.04
5,151,684	9/1992	Johnsen	340/572
5,153,583	10/1992	Murdoch	340/825.54
5,223,844	6/1993	Mansell et al.	342/357
5,347,274	9/1994	Hassett	340/988
5,469,363	11/1995	Saliga	364/478.13
5,517,194	5/1996	Carroll et al.	340/825.54 X
5,521,602	5/1996	Carroll et al.	340/825.54 X
5,528,232	6/1996	Verma et al.	340/825.54
5,546,540	8/1996	White	395/200.1

OTHER PUBLICATIONS

"Rafio tracking/ID & security system", Savi Technology, Commercial Carrier Journal, p. 122, Aug. 1995.

The Savi Asset Management System™, Savi Technology, Inc., Palo Alto, CA, No date.

TransitMaster™An Integrated, GPS–Based Information Management System for Transit, Rockwell International, Cedar Rapids, IA, Aug. 1995.

"TMS Wins Vehicle Location Contract for Des Moines Metro", Earth Observation Magazine, p. 12, Oct. 1995.

"Will TI make a Bundle on 'Smart' packages?", Business Week, p. 112, Nov. 27, 1995.

Varon, E., "Smart Cards Tech may keep drivers truckin'" No source or date.

ProfitMAX™ Highlights, Integrated Cargo Management Systems, Inc., 3 pages, No date.

"Auto–Trac Provides the Information You Need", Auto–Trac, Dallas, TX, 4 pages, No date.

3M Fleet Management System Pinpoints Bus Activity, 3M, 2 pages, No date.

Okabe et al., "Spatial Tesselations Concepts and Applications of Voronoi Diagrams", chapters 1–4, pp. 1–10, 65–121, 209–271. No date.

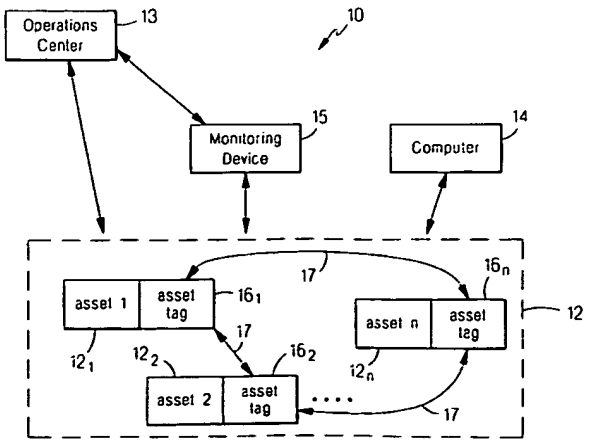
Primary Examiner—Edward R. Cosimano

Attorney, Agent, or Firm—Fish & Richardson P.C.

[57] **ABSTRACT**

An object in a storage area or moving vehicle is monitored by attaching an electronic tag to the object. An electronic device detects the presence of the object by communicating with the tag while the object is in storage or is being moved by the vehicle. The tags may also determine the location of an attached object and may reroute the object if it deviates from a given shipping schedule. A group of objects is monitored by two electronic tags, each attached to an object in the group. Each tag has circuitry for communicating information relating to an object in the group to a second tag. Each tag also includes a memory connected to the circuitry that is capable of storing the information, and a controller connected to the memory and the circuitry.

60 Claims, 59 Drawing Sheets



include an extra delay which is a multiple of the second clock period; and the distance may be determined by multiplying the velocity by half of the difference between (a) the average of all of the round-trip times and (b) the sum of the extra delay and the first clock period. The method may further include a step of offsetting the average of the round-trip times, wherein the offset is an amount of time equal to the second clock period divided by twice the product of (a) the number of symbols and (b) the number of first clock periods corresponding to the first time interval.

In another aspect, the invention features another method of measuring a distance between two objects. The method includes the steps of transmitting from one of the objects a plurality of symbols, the transmissions occurring in a series in which consecutive transmissions are separated by a time interval which is a multiple of a first clock period, the symbols having a constant velocity; detecting the transmissions using a second clock period, the second clock period being different from the first clock period, each transmission detection thus occurring after a delay, the delay being different for each detection and constituting the period of time between each transmission and the detection of each transmission, the delay having a nominal length which is a multiple of a delay unit, the delay unit being equal to the product of the number of first clock periods corresponding to the time interval and the difference between the clock periods, the delay unit also being equal to the second clock period divided by the number of symbols in the plurality of symbols, the delay having an actual length which is within one delay unit of the nominal length; returning each of the symbols from the other of the objects back to the one of the objects; detecting the arrivals of the returning symbols back at the one of the objects using the second clock period, each arrival detection thus occurring after a second delay, the second delay being different for each detection and constituting the period of time between each arrival and the detection of each arrival, the nominal length of the second delay being a multiple of the delay unit, the second delay having an actual length which is within one delay unit of the nominal length of the second delay; measuring a round-trip time corresponding to detection of receipt from the other of the objects of each of the symbols, each round-trip time corresponding to a multiple of a second clock period; and determining the distance between the objects by multiplying the velocity by half of the average of the round-trip times.

A preferred embodiment of this aspect of the invention includes the following feature. The symbols may be returned from the other of the objects after an extra delay; and the distance may be determined by multiplying the velocity by half of the difference between the average of all of the round-trip times and the extra delay.

Advantages of the invention include the following.

An operator is required to enter data into the system only once. After entry, the tags and devices in the system automatically move the data to where it is most needed. For example, the operator enters the contents of a particular asset into the tag attached to the asset. The tag then automatically communicates the contents to an inventory computer storing a list of local assets.

Because the tags communicate with other tags in their vicinity, the effective communications range of each tag extends beyond the range of its own communication circuitry. The effective range is further extended by placing an additional tag near a selected tag in the system. The selected tag relays communications from the remaining tags in the system to the additional tag. The remaining tags are thus not required to be located within the communication range and line of sight of the additional tag.

Once the tags are activated, they automatically seek other tags or devices in order to report their status. The system thus does not need to depend on the honesty of company employees in collecting and reporting data relating to the assets. The information reported by the tags takes a variety of forms. For example, a tag may monitor a sensor for the temperature of an object, and may notify a device of an abnormal increase in temperature. Similarly, the tag may determine a scheduled route for the asset, and report any deviations from that route to the user-operated device.

An operator obtains information relating to an asset in the system by directly querying an appropriate tag. Because several tags in the system share the information and can automatically move the information, the operator is likely to have access to a tag that either stores the needed information or that can acquire the information. The operator is thus not required to access an operations center that may be outside the operator's communications range in order to obtain the information.

A local or remote computer is easily integrated into the system. The computer gathers and disseminates information to operators who do not possess the appropriate device for querying tags, or to operators who are located far from any tag. The computer maintains a list of tags, their location and associated asset, so that any operator query can be directed to the tag.

Unlike the barrier systems described above, the system provides continuous spatial monitoring of the assets. Instead of only determining that a particular asset is located anywhere inside one or more cells, the system accurately locates a position of the asset within any cell. In addition, while barrier systems cannot typically determine a direction of movement of an asset when it crosses a boundary, the system according to the invention accurately determines direction and velocity of the asset.

Knowing the position and velocity of the asset allows operators to promptly find the asset in response to an emergency, and permits accurate policing of the asset to prevent future incidents. The assets are thus more secure from theft or misplacement, which is an important consideration when the assets include valuable cargo. In addition, the quick containment of incidents provides for more safety in transporting and storing of the assets. This is particularly important for assets carrying hazardous materials.

In some embodiments, the tags form an optimal communications network without user input, and update the network whenever there is a change in the location or identity of the assets. The tags define the network by designating tags for relaying information between specific pairs of tags. The network is optimized by minimizing the number of relay tags between every pair while maximizing the range of the communications. The tags superimpose on this network a hierarchy of communication links to other networks or devices.

The tags also optimize the network by assigning specific functions to particular tags in order to prevent tags from needlessly repeating the functions of other tags. These functions include checking whether assets have been added to or removed from the network.

The tags design the network based on their calculations of the location of other tags. Error correcting algorithms in the tags permit them to compute these locations with a high degree of accuracy. When new information becomes available to a tag (for example, when a new tag is added to the network), the tags automatically refine their location calculations.

the entities is listed in the entity table (step **468**), the attachment unit becomes inactive (step **469**).

The tag also notifies the joining unit of the detachment in step **464**. Alternatively, the subfunction may cause the router to notify a device, such as a user-operated device, instead of a gateway of the detachment.

(b) Joining

The joining functional unit **372** includes an existence subfunction **470** connected below a hierarchy subfunction **472** (FIG. **24**).

Referring to FIG. **29**, when the attachment subfunction indicates to the existence subfunction that the tag is active (step **480**), the existence subfunction causes the tag to look for other tags by attempting to initiate communications through the router on a particular hailing channel (**482**). If other tags exist (step **484**), the subfunction notifies the hierarchy and supplies the identity codes of the tags (step **486**).

If a group of tags are located near the asset tag, the subfunction chooses one asset tag (a netmaster) to perform some of the location and network formation units. This prevents the tags from duplicating calculations already performed by other tags.

Referring to FIG. **30**, the hierarchy subfunction communicates with the other tags to determine if the other tags have designated a particular tag as netmaster (step **487**). If the tag locates a netmaster (step **488**), it notifies the network formation unit through the location unit (step **489**). Otherwise, the tag chooses the tag with the highest identification number as the netmaster, and chooses the tag with the next highest identification number as the assistant netmaster (step **490**).

The netmaster periodically surveys the tags for new or missing members, and notifies the hierarchy subfunction. The netmaster uses a particular hailing frequency for its communications.

If the netmaster disappears or indicates its battery is low, the assistant netmaster replaces the netmaster, selects another assistant, and informs the hierarchy subfunction of the change.

(c) Location

Referring to FIG. **25**, the location unit **374** determines the locations of tags in the network using the following subfunctions: intertag distance **500**, verify range **502**, refine range **504**, location compute **506** and detect change **508** (FIG. **25**).

Referring to FIG. **31**, the intertag distance subfunction **500** determines the distance and the covariance (or measurement error) of the distance between the tag and each tag identified by the joining unit (step **530**). The subfunction also communicates with the tags to obtain location information for the tags (step **532**). The location information is typically obtained from a GPS sensor located near the tags. The subfunction organizes this information in the tag memory (step **534**), and notifies the verify range subfunction **502** that the information is available (step **536**).

The verify range subfunction verifies these ranges, and eliminates ranges that are clearly erroneous. The refine range subfunction provides accurate estimates for the ranges eliminated in the previous subfunction. The compute locations subfunction transforms the corrected ranges supplied by the refine range function into an arbitrary set of coordinates for each tag. Using the location information, the location subfunction transforms the arbitrary coordinate system into an absolute coordinate system, and determines a location for the tag.

The detect change subfunction receives the location information from the verify range subfunction, and monitors the ranges of other tags in the network to detect a change in the tag's location.

Each of these subfunctions is discussed in more detail below.

(d) Network Formation

The network formation unit includes a determine topology subfunction **560**, a sentinel subfunction **562** and a message subfunction **564** (FIG. **25**). The determine topology subfunction receives information from the change detection subfunction indicating changes in the existence of a tag, and the location of the tag over time. Using this information, the subfunction determines a set of neighbors for each tag. A topology description of the set is found using a Delaunay triangulation, described below, which describes the topology as a convex hull and identifies the set of nearest neighbors. When the determine topology subfunction receives updated information from the location unit, it modifies the topology description.

The sentinel subfunction identifies tags that are located on the convex hull. These tags (referred to as sentinels) are assigned to communicate with tags outside the hull. The subfunction also identifies gateway tags capable of long distance communication and PDAs acting as a man-machine interface. The subfunction stores the identity of each gateway in the gateway table.

The sentinel subfunction communicates the identity of the sentinels to the hierarchy subfunction **472** (FIG. **24**) over the feedback path **380**. The hierarchy subfunction causes the sentinel to search for new tags, extending the range of the netmaster.

The sentinel subfunction also responds to commands received over the router **382** (FIG. **23**). Each time a command from an entity is routed to the sentinel subfunction, the subfunction first verifies that the source of the message is listed in its entity table. If the entity is not listed, the subfunction ignores the message. If it is listed, the unit responds to the command.

The first command received by the sentinel subfunction queries whether a tag with a given identification exists. The subfunction checks whether any stored identification matches any stored tag's identification number. If it does, it answers "yes" to the entity. The unit then checks whether the entity is listed as a gateway in the gateway table. If it is not listed, the unit records the identity of the entity as a gateway. If there is insufficient memory to store the identity in the table, the unit communicates with the oldest recorded gateway in the list. If it is unable to establish communication, it deletes the gateway from the list. The unit repeats this for each gateway in the list.

The second command asks whether there are tags which have not been queried by a gateway. The subfunction looks at its gateway table, and if the gateway ID is not listed, the unit replies with the tag ID.

The last command asks whether a tag with a particular ID has registered the gateway. If the tag ID is the same as that in the message, the unit checks its entity and gateway table and records the ID of the gateway if it is not already present in the table. If there is insufficient memory to store the gateway, the tag repeats the process described above to delete non-communicating gateways.

The propagate message subfunction defines a network containing a set of efficient communication links between tags using the topology derived by the topology subfunction. The subfunction accomplishes this by assigning tags in the networks as relay points for communications between a pair of tags.

The subfunction also chooses a second netmaster based on the topology. The second netmaster redefines the network whenever a tag is added or removed from the network, and

communicates information regarding the changes in the network to the gateways listed in the gateway table. In some embodiments, the netmaster only notifies the gateway of changes regarding specific tags. The other tags in the network do not duplicate the functions of the netmaster.

The netmaster polls members of the network periodically to make sure they still exist. Each tag in the network checks the time interval since it was last polled by the netmaster. If the time interval exceeds a threshold, the network formation unit in the tag removes the tag from the network.

When the network grows too large, the network formation subfunction divides the network into smaller subnetworks. The subfunction designates channels of communication within a subnetwork that do not interfere with communications in the other subnetwork. In addition, communications within a subnetwork are coded in compacted form. For example, the subfunction assigns each tag a truncated ID. If spread spectrum communications are used, the subfunction uses a short code for communication relayed between tags.

The subfunction designates a tag for maintaining communication with another subnetwork. When the subnetwork grows too small, the subfunction queries the tag for the existence of the other subnetwork. If the subnetwork still exists, the subfunction merges the two subnetworks.

(e) Network Use

The network use unit **378** includes a manager subfunction **590**, an agent subfunction **592**, an inventory subfunction **594**, and a read and write subfunction **596** (FIG. **25**).

Referring to FIG. **32**, the manager subfunction continually checks whether route information is available in the network (step **608**). The information is typically provided by a monitoring device, computer or operations center.

When the information is available, the subfunction stores information regarding the transportation route scheduled for an asset attached to the tag and a checking frequency (step **610**). The subfunction then surveys the network to determine whether there has been an update to the route information (step **611**). If there is, the subfunction returns to step **610** and stores the new information.

If no update information is available, the subfunction examines the location of the tag (as supplied by the location unit) (step **612**), and compares the location to the route (step **614**). If the location of the asset deviates from the route (step **616**), the subfunction informs a gateway through the network (step **618**) and returns to step **611**. If no deviation exists, the subfunction repeats the process by returning to step **611**.

As shown in FIG. **33**, the agent subfunction **592** stores information regarding the destination of the attached asset (step **630**). When a deviation from the route is reported by the manager subfunction (step **632**), the agent subfunction queries the network regarding available alternative routes (step **634**). If a more effective alternative route is available (step **636**), the subfunction notifies a gateway (step **638**).

In some embodiments, the route includes a series of locations where the conveyance carrying the asset is changed (transhipment points). The subfunction seeks a conveyance for the next part of the route at each transhipment point. The subfunction also searches for other tags following a similar route, and searches for a proper conveyance for carrying all the assets together. The subfunction then informs a tag on the desired conveyance that the conveyance should load the assets. The subfunction thus performs functions analogous to a node in a packet switched network.

The inventory subfunction maintains a list of assets in the network. This subfunction is particularly useful in a warehouse application.

The read and write subfunction stores and retrieves data from the asset tag's memory. The subfunction is activated by the following commands from one or more entities stored in the entity list.

1. Read data to one or more identified entities.
2. Clear data as instructed by one or more identified entities, and write certain data.
3. Append the following data as instructed by one or more identified entities.
4. Add an identified entity to the entity table as instructed by one or more entities.
5. Delete an entity from the entity table.
6. Read the entity table.
7. Read the gateway table.

X. Detailed design of the location unit

A. Intertag Distances Subfunction

(a) Baseline Method.

The intertag distances subfunction (FIG. **34**) in a tag **640** determines the tag's distance, that is, range, from another tag **642** by measuring the time required for an electromagnetic pulse to travel round-trip between the two tags. There are several methods for measuring this time. A baseline method is described first, followed by descriptions of alternate methods. Most of the methods require that the pulse-originator tag have a cooperating transponder tag for a reference, with one clock on each tag. However, the last alternate method describes a dual-clock technique wherein both clocks are on the originator tag—the other tag is simply a repeater or a repeater with a delay. In fact, if a delay is not required, another active tag is completely unnecessary in the last alternate method; the reference can be totally passive, for example, a mirror or a three-dimensional corner reflector.

The resolution of the range measurement is proportional to the time-measuring resolution of the tag, so it is advantageous to use methods of range measurement that effectively increase the time-measuring resolution. The methods described below take advantage of the tag's ability to detect a pulse received during a period of the tag's time-measuring clock. For a single pulse, the tag can detect that the pulse arrived during the period but not when during the period. Therefore, the range-measurement methods use multiple pulses and multiple successive measurements. In the baseline method, small offset delays of differing but known lengths are inserted, one per round-trip, in the round-trips of the pulses, to stagger the arrivals of the pulses. Thus, for a set of ranges, the offset delays cause a fraction of the pulses to return in a later time-measuring clock period in any particular range measurement. As described below, the method determines the range from this fraction.

In the baseline method, the originator tag and the transponder tag initially have identical clock periods. To insert a multiple of an offset delay "d" in the round-trips (FIG. **35**), the originator alters its clock period slightly. The length of the alteration is an amount of time which provides a one-clock-period difference with the transponder's clock over a measurement interval. The measurement interval is an amount of time equal to the number of pulses sent, "N," multiplied by the time interval between the sending of each pulse. Although the originator's clock may be altered above or below the frequency of the transponder's clock, the embodiment of the baseline method described herein uses an originator's clock with a higher frequency, that is, a shorter period, than the transponder's clock.

The difference in clock periods results in the insertion of differing but known delays in the round-trip time, in the following way. Like the originator tag, the transponder tag

If the tag and reference tags are coplanar but not co-linear, the reference tags must be moving in two dimensions relative to the tag. If the tag and reference tags are co-linear, the reference tags must be moving in one dimension relative to the tag.

As additional reference positions and errors are accumulated by the intertag distance subfunction, the tag refines its position estimate using the same method. In addition, as tag locations are identified and changed, this information is passed to the determine topology subfunction.

XI. Detailed description of network formation unit

A. Determine topology subfunction

The determine topology function designates a set of neighbors for each tag by means of Delaunay triangulation, described in detail in "Spatial Tessellations: Concepts and Applications of Voronoi Diagrams," A. Okabe et al, the contents of which are incorporated herein, in its entirety, by reference.

Referring to FIG. **53**, the neighbors of a tag 16_1 are found by first locating the set of points that are closer (in Euclidean distance) to tag 16_1 than to any other tag. Each set of points is bounded by an "ordinary Voronoi polygon" **2000**. The neighbors of each tag include the tags in polygons adjacent to polygon **2000**. Thus, the neighbors of tag 16_1 consist of tags $16_2, \ldots, 16_8$.

Referring to FIG. **54**, a preliminary communication link **2010** is defined between a tag and each neighboring tag. The set of links between tags forms a Delaunay triangulation, and defines the topology of the set of tags.

The Delaunay triangulation is defined incrementally by causing each tag to find its neighbors. If the tags are all activated at the same time, the netmaster (the tag with the highest ID number) finds its neighbors first, by using the algorithm described below. After the netmaster has completed the algorithm, it selects the tag having the second highest ID number, and commands the tag to perform the algorithm to find its neighbors. The netmaster continues selecting tags in decreasing order of ID number until all tags have been selected.

When a new tag is added to a network, it communicates with other tags to determine whether the tags have already performed the process described above, that is, whether a Delaunay triangulation for the tags has been determined. If it has, links must be added to the triangulation to connect the new tag to its neighbors. This is accomplished by having the new tag perform the algorithm described below. If no Delaunay triangulation exists, the new tag waits until ordered by the netmaster to perform the algorithm.

The new tag (or the tag designated by the netmaster) finds and maintains a list of its neighbors by performing the following algorithm. Each step of the algorithm is executed by the determine topology subfunction in the new tag.

The new tag begins by receiving its own coordinates and then seeks information locating other tags in its vicinity from the compute locations subfunction. The determine topology subfunction ignores the z coordinates in the location information and thus finds the topology of the tags using a two-dimensional projection of their locations.

The determine topology subfunction next selects a random tag from the coordinate information which it defines as the "current tag" (current(n)) where n is the index of the iteration and queries the current tag regarding its neighbors. In other embodiments, the tag may communicate with a sentinel (defined in Section IX.C(c), above) that causes the tag to choose the sentinel as the current tag. Any other tag may be chosen as the current tag.

Referring to FIG. **55**, the new tag calculates its distance from the current tag and its distance from each neighbor of the current tag, using the coordinate information (step **2014**). The new tag analyzes the distances to determine whether the new tag is closest to the current tag (steps **2016, 2018**). If so, the new tag stores the identification number of the current tag in its list of neighbors (step **2020**) and proceeds to step **2022** in FIG. **56**. If not, the new tag chooses the neighbor of the current tag that is nearest the new tag as the current tag (step **2022**) and returns to step **2014**.

When the new tag arrives at step **2022**, the current tag has been defined as either the last current tag or the neighbor of the last current tag that is closest to the new tag. In step **2024** (FIG. **56**), the new tag defines two of the neighbors of the current tag as the "next tag" and the "previous tag."

The next tag of a particular tag A is defined relative to a second tag B. The next tag is the neighbor of tag A which forms the smallest positive angle c with respect to tag B, with the tag A being placed at the vertex of the angle. For example, as shown in FIG. **57**, tag A is the current tag 16_1 and the tag B is the new tag 16_2. If the neighbors of the current tag include tags 16_3, 16_4 and 16_5, then the next tag of the current tag relative to the new tag of the current tag is tag 16_3. Similarly, the next tag relative to tag 16_3 is tag 16_4.

The previous tag of a current tag relative to a second tag is defined as follows. The previous tag is the neighbor of the current tag which forms the smallest negative angle d with respect to the second tag, with current tag placed at the vertex of the angle. In the example shown, the previous tag for the current tag 16_1 with respect to the new tag 16_2 is the tag 16_4. Similarly, tag 16_3 is the previous tag for the current tag relative to tag 16_4.

The new tag defines the next tag (next(n)) and the previous tag (previous(n)) of the current tag relative to the new tag (step **2024**). The new tag then checks whether the angle between the previous tag and the next tag (with the current tag at the vertex of the angle) is less than 180 degrees (step **2026**). For example, in FIG. **57**, the new tag checks whether the angle d is less than 180 degrees. In the same step, the new tag verifies that the previous and next tags are neighbors of one another (that is, they are connected in the Delaunay triangulation). The new tag accomplishes this by querying the previous tag and the next tag regarding their neighbors.

If both conditions of step **2026** are satisfied, the new tag checks whether the new tag is located inside a circle circumscribing the current, next and previous tags (step **2028**). For example, as shown in FIG.. **58**, circle **2029** circumscribes tags 16_1, 16_3, 16_4 and contains the new tag 16_2. If the condition of step **2028** is satisfied, a new previous tag (previous (n+1)) is defined as the next tag (next (n)), and a new next tag (next (n+1)) is defined as the tag having the smallest positive angle with respect to the next tag (next (n)). The new tag then increments the value of the iteration (n=n+1) and returns to step **2026**.

If the condition in step **2026** is not satisfied, the new tag checks whether the new tag is between the previous tag and the next tag with respect to the current tag (step **2034**). This means that the previous and next tags are previous and next tags relative to the new tag. For example, in FIG. **57**, the new tag 16_2 is located between the previous tag 16_5 and the next tag 16_3.

If the condition in step **2034** is satisfied, the new tag adds the previous tag to its list of neighbors, and informs the previous tag (steps **2028–2032**). The previous tag then adds the new tag to its list of neighbors (step **2036**). Similarly, if the condition in step **2028** is not satisfied, the new tag proceeds to step **2036**.

After step **2036**, the new tag checks whether the previous tag is identical to the closest tag found in step **2018** (step **2037**). If so, the algorithm skips to step **2038** (FIG. **60**). If not, the new tag checks whether the angle of the new tag to the previous tag (with the current tag at the vertex of the angle) is greater than or equal to 180 degrees (step **2040**). For example, in FIG. **57**, the new tag compares the angle d to 180 degrees. If the condition is not satisfied, the current tag (current(n+1)) is redefined as the previous tag (previous (n)) (step **2042**), the index n of the iteration is incremented by one (step **2044**), and the new tag goes to step **2024**. If the condition in step **2040** is satisfied, the algorithm proceeds to step **2022'** (FIG. **59**).

Referring to FIG. **59**, in step **2022'**, the algorithm begins analyzing the previous and next tags in an opposite order. Step **2022'** is identical to step **2022** in FIG. **56**. The algorithm next repeats steps **2024** to **2044**, with the following changes. In step **2030'** (which replaces step **2030**), a new next tag (next (n+1)) is defined as the previous tag (previous(n)). A new previous tag (previous(n+1)) is defined as the tag forming the smallest negative angle (with the current tag at the vertex of the angle) relative to the previous tag (previous (n)).

In step **2036'** (which replaces step **2036**), the next tag (next (n)) is added to the list of neighbors of the new tag. The new tag then informs the next tag, which adds the new tag to its list of neighbors. Steps **2037** and **2040** are replaced with step **2037'**, in which the new tag checks whether the next tag is the closest tag, or the angle between the new tag and the next tag (with the current tag at the vertex of the angle) is greater than or equal to 180 degrees. If the condition is not satisfied, the algorithm proceeds to step **2042**. If the condition is satisfied, the algorithm proceeds to step **2038** (FIG. **60**).

Referring to FIG. **60**, in step **2038**, the new tag finds a pair of its neighbors (A, B) that are neighbors of each other, but are not consecutive with respect to the new tag. This means that tag A is not the neighbor of the new tag that forms the smallest positive angle with tag B, with the new tag placed at the vertex of the angle. Similarly, tag A is not the neighbor of the new tag that forms the smallest negative angle with tag B, with the new tag placed at the vertex of the angle. FIG. **61** shows a situation where the condition is satisfied, with tags A, B and C as neighbors of new tag 16_2.

In step **2038**, the new tag also checks that tags A and B form an angle smaller than 180 degrees (with the new tag placed at the vertex of the angle). In the example shown in FIG. **61**, the new tag compares angle e to 180 degrees. The algorithm checks every possible pair of neighbors, and if no pair of neighbors satisfy the conditions in step **2038**, the algorithm ends. For each pair satisfying the conditions of step **2038**, the new tag checks whether one of the following two conditions are satisfied: (1) the next neighbor of tag A relative to tag B, and the previous neighbor of tag B relative to tag A are the same tag (tag C), and (2) the next neighbor of tag B relative to tag A and the previous neighbor of tag A relative to tag B are the same tag (tag C) (step **2050**). If either condition is satisfied, the new tag checks whether tag C is not the new tag and the new tag is included in a circle circumscribing tags A, B and C (step **2052**). If the condition is satisfied, the tag disconnects tags A and B, that is, the new tag informs tags A and B that they are not neighbors of one another, and tags A and B modify their lists of neighbors accordingly (step **2054**). The new tag then returns to step **2038**.

If the condition in step **2052** is not satisfied, the algorithm returns to step **2038**. If the condition in step **2050** is not satisfied, the new tag checks whether the angle of the next tag of tag A relative to tag B (with the vertex of the angle at tag A) or whether the angle of the next tag of tag B relative to tag A (with tag B at the vertex of the angle) is less than 180 degrees (step **2056**). If so, the new tag proceeds to step **2054**; if not, the algorithm returns to step **2038**.

When a tag is removed or deleted, the netmaster directs updates of the topology by having the following algorithm performed by the neighbors of the removed tag. Each step of the algorithm is performed by the determine topology subfunction in the netmaster.

Referring to FIG. **62**, the netmaster identifies a current tag as a neighbor of the removed tag (step **2100**) and directs it to execute the following. The next tag (next(n)) is defined as the neighbor of the removed tag that is the next tag relative to the current tag with respect to the deleted tag. The previous tag (previous(n)) is defined as the neighbor of the removed tag that is the previous tag relative to the current tag with respect to the deleted tag (step **2102**).

The tag checks whether the next tag is the same as the previous tag and whether the previous and next tags are neighbors of one another (step **2104**). If either or both conditions are satisfied, the algorithm causes each neighbor of the deleted tag to remove the deleted from its list of neighbors. The deleted tag is thus disconnected from all its neighbors, and the process ends.

If the conditions of step **2104** are not satisfied, the netmaster checks whether the current tag is connected to the next and the previous tags, that is, the current, previous, and next tags are neighbors of one another (step **2106**). If not, the tag defines a new current tag (current(n+1)) as the next tag (next(n)), increments the index n of the iteration (step **2108**) and returns to step **2102**. The removed tag completes the process by removing all the neighbors from its list.

If the condition in step **2106** is satisfied, the tag checks whether a neighbor of the deleted tag other than the current, next or previous tags is contained in a circle circumscribing the next, current and previous tags (step **2110**). If the condition is satisfied, the algorithm returns to step **2108**. If not, the tag checks whether the deleted tag is contained in the circle (step **2112**). If the condition is not satisfied, the algorithm returns to step **2108**. If the condition is satisfied, the deleted tag is removed from the list of neighbors of the current tag, and the next and previous tags are listed as neighbors of one another (step **2114**). The algorithm then returns to step **2108**.

When a tag is moved from one location to another, the tags update the topology by performing the algorithm illustrated in FIG. **62**, in which the moved tag is defined as the deleted tag. The moved tag then finds its neighbors in its new location by performing the algorithm illustrated in FIGS. **55**, **56**, **59** and **60**.

B. Propagate messages

The propagate messages subfunction **564** (FIG. **25**) defines a set of relay points for efficient communications between tags, and defines a hierarchy of networks.

Each network within a level of the hierarchy is defined by the physical constraints of the system. For example, in one type of system, the assets include items placed in a container. Each container is in turn stored on a pallet carried by a vehicle. An asset tag is attached to each item, pallet, container and vehicle in the system. The set of tags on the items within a container neighbor one another, and automatically form a network. Because the tag on the container is likely to be within the communication range of only a few tags on the items, the tag on the container is generally not included in the network. Similarly, the tags on the packages, pallets,